International Journal of Soft Computing 9 (5): 318-332, 2014
ISSN: 1816-9503

A Cylindrical Coordinate System with Dynamic Permutation Table for Blowfish Algorithm

[1, 2]Ashwak Alabaichi, [3]Ramlan Mahmod and [1]Faudziah Ahmad
[1]Department of Information Technology, University Utara Malaysia,
Kedah, 06010 Sintok, Malaysia
[2]Department of Computer Science, Faculty of Sciences, Kerbala University, Karbala, Iraq
[3]Faculty of Computer Science and Information Technology, University Putra Malaysia,
Serdang, 43400 Selangor, Malaysia

Abstract: The Blowfish Algorithm (BA) is a symmetric block cipher that iterates simple encryption and decryption functions by using Feistel networks. BA keys vary from 32-448 bits to ensure a high level of security. However, the BA requires a high memory percentage and it has a problem regarding randomness of output with text and image files having large strings of identical bytes. One solution to the seissues is to design a new Cryptography algorithm based on the BA that incorporates an F-function into a Cylindrical Coordinate System (CCS). The resulting F-function is known as a CCS with a Dynamic Permutation Table (DPT) or CCSDPT whereas the new algorithm is called the New BA (NBA). The objectives of the CCSDPT are to reduce memory requirements, enhance the randomness of the output and increase resistance to attacks through byte relocation and transformation in the right cylinder. NBA is evaluated by investigates the output of the algorithm by using statistical tests from the National Institute of Standard and Technology (NIST) with five types of data and compared with the BA. The findings of the NIST tests show that the NBA is suitable for any data stream, even those with long strings of identical bytes. The combination of a DPT with a dynamic 3D S-box strengthens the resistance of the NBA against attacks and increases the randomness of the output. C++ is used in the implementation of both algorithms. The NIST tests are implemented under Linux.

Key words: Blowfish algorithm, F-function, cylindrical coordinate system, dynamic permutation table, byte permutation, byte relocation, byte transformation, right cylinder

INTRODUCTION

The Blowfish Algorithm (BA) is a secret-key block cipher proposed by Schneier (1994). It is a Feistel Network that can iterate a simple encryption function 16 times. The block size is 8 bytes and the maximum key up to length is 56 bytes (Zhang and Chen, 2008).

The BA dynamically computes all subkeys and S-boxes before the start of encryption thus resulting in a slight overhead in computation. This computational overhead.

From subkeys and S-boxes results in nearly the equivalent of encrypting an additional 4 kB of data per data file. This large memory requirement makes the BA infeasible for smart card applications and embedded systems (Zhang and Chen, 2008; Mahdi, 2009; Wang *et al*., 2011; Chandrasekaran *et al*., 2011).

Another issue with the BA lies in its incompatibility with image and text files that involve large strings of identical bytes, particularly problems related to the randomness of the output with encrypted text and image files (Alabaichi *et al*., 2013a, b).

In this study, researchers propose a Feistel network, 128 bit block cipher for the BA that includes a new F-function based on a Cylindrical Coordinate System (CCS) with a Dynamic Permutation Table (DPT) or CCSDPT by using a variable key length ranging from 8 bytes up to 64 bytes and iterating encryption and decryption functions 10 times. The advantages of the CCSDPT include reducing memory requirements and improving security with 10 rounds. The security of the CCSDPT is increased by several techniques such as byte relocation based on secret keys, byte transformation based on secret keys for sections of the right cylinder in three different cases and a permutation table based on secret keys. Thus, a secure CCSDPT leads to a secure New BA (NBA) 128 bit.

Corresponding Author: Ashwak Alabaichi, Department of Information Technology, University Utara Malaysia, Kedah, 06010 Sintok, Malaysia

In the new F-function (that is, CCSDPT), memory requirements are reduced by using the CCS as a single S-box in three dimensions. The size of the 3D S-box is 256 bytes. The size of four S-boxes is 4096 bytes in the BA and 2,097, 152 bytes ($2^{16}\times8\times4$) in the 128 bit extension BA. Researchers use the same outer structure of the BA in the extension procedurefor the 128 bit BA (Hashim *et al*., 2009; Alabaichi *et al*., 2013b). Cryptographic strength is defined in this study as the ability of the algorithm to produce a random output. The experimental resultsfrom the randomness test on the five data categories showed that the NBA 128 bit is better than theBA and that the former is compatibleto all types of files without restrictions on file contents.

Researchers designed the CCSDPT based on secret keys. It includes a dynamic 3D S-box and a DPT that vary in each round along the data stream. That is, the dynamic 3D S-box and the DPT vary during the encryption process in each round. The CCSDPT is designed to satisfy the confusion and diffusion principles suggested by Shannon (1949) by using the CCS as a 3D S-box and by applying byte permutation to diffuse the elements of the 3D S-box through two procedures based on random secret key sand the DPT based on secret keys. The CCSDPT is significant because it reduces memory requirement, enhances randomness of the NBA128 bit and increase security against differential and linear attacks.

Researchers used the test suite (Bassham *et al*., 2010) from the National Institute of Standard and Technology (NIST) to analyze the sequences generated by the NBA 128 bit. These statistical tests are suitable for evaluating random number generators and pseudo-random number generators used in cryptographic applications.

BLOWFISH ALGORITHM (BA)

The BA consists of two parts: a key-expansion part and a data-encryption part. Key expansion converts a key with at most 448 bit into several subkey arrays that total 4168 bytes. Data encryption occurs via a 16 round Feistel network. Each round consists of a key-dependent permutation as well as a key and a data dependent substitution. All operations are exclusive ORS (XORs) and additions on 32 bit words. The only additional operations are four indexed array data lookups per round. The F-function hasan important role in the original algorithm that is, it splits the 32 bit input into four 8 bit quarters and then uses these quarters as inputs for the S-boxes. The outputs are added modulo 2^{32} and XORed to produce the final 32 bit output. The F-function is defined as follows: $F(X) = \{[(S_1+S_2) \bmod 2^{32} \oplus S_3]+ S_4\} \bmod 2^{32}$.

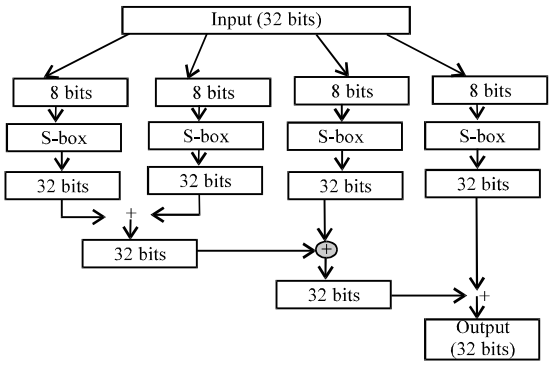


Fig. 1: The F-function architecture

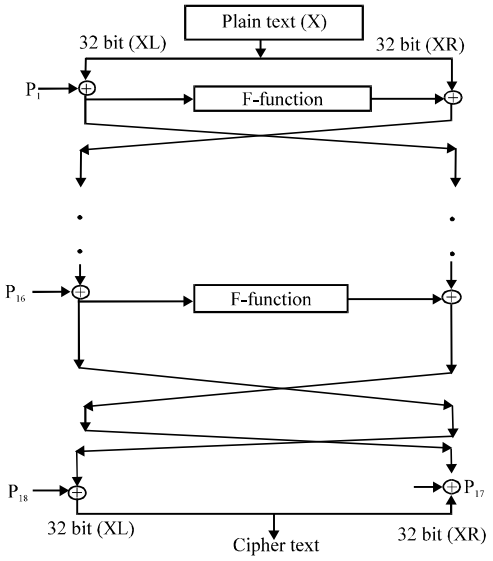


Fig. 2: The BA architecture

The F-function and the architecture of the BA are described in Fig. 1 and 2, respectively (Al-Neaimi and Hassan, 2011; Manikandan *et al*., 2012a, b; Schneier, 1994; Cornwell, 2012).

Nechvatal *et al*. (2000) stated that a 128 bit input is the minimum required block size. The strength of the symmetric key encryption depends on the size of the key used. For the same algorithm, the encryption that uses a longer key is harder to break than the one that uses a shorter key (Nie and Zhang, 2009). Therefore, the NBA is 128 bit and has a key length of 64 bytes instead of 56 bytes without any increase in memory requirement. The NBA 128 bit is described in study.

THE NBA 128 BIT

The NBA128 bit which is a Feistel network similar to BA, consists of two parts: a key-expansion part and a data-encryption part.

Researchers propose a new function (CCSDPT) that includes four key-dependent 3D S-boxes and one key-dependent permutation table per round of the encryption process to thwart all linear and differential trails from attackers completely. The first 3D S-boxis generated by applying the first byte permutation procedure called byte relocation (this process is explained later). The three other 3D S-boxes are generated by the second byte permutation procedure called byte transformation in the right cylinder. This process enables the cipher to deny any attack based on the specific properties of the S-box.

Key expansion: Key expansion converts a variable-length key of up to 64 bytes into subkey arrays that total 352 bytes. A large number of subkeys are used as indicated:

- The P-array of 1264 bit subkeys: P_0, P_1, ..., P_{11}
- One 8 bit 3D S-box has the following entries: 0, ..., 7, 0, …, 7, 0, ..., 3 (this process is illustrated later)
- Researchers use five sets of secret random numbers keys generated from random function along the encryption process (this process is illustrated later)
- Researchers use other secret keys with transformations in the right cylinder and permutation table (this process is illustrated later)

The NBA 128 bit has numerous advantages, one of which is the use of a large number of keys thus increasing the key space of secret keys. A large key space is necessary to prevent exhaustive searches for a key (finding the correct value for a key by testing all possible values until the correct one is found).

The same procedure used to generate subkeys in the previous BA is used in the NBA 128 bit to generate these subkeys (p_0, ..., p_{11}) and the 3D S-box because as Schneier (1994) stated this procedure preserves the entire entropy of the key and distribute the entropy uniformly throughout the subkeys. This procedure is designed to distribute the set of subkeys randomly throughout the domain of possible subkeys.

Data encryption: The Encryption algorithm of the NBA 128 bit is similar to that of the BA. The input of the NBA 128 bit is 128 bit data plaintext P and the output is 128 bit cipher text C. The external structure however is the same as that of BA. The input is 128 bit which is divided into 64 bit halves XL and XR. Algorithm 1 of the encryption NBA 128 bit is as follows:

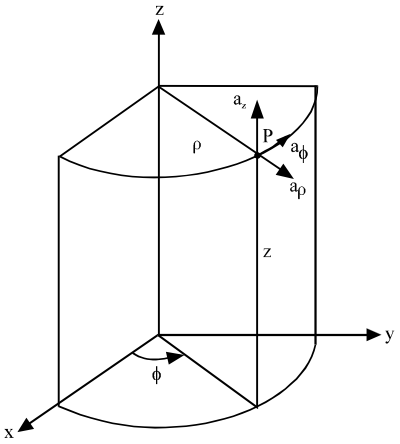


Fig. 3: Cylinder coordinates system

Algorithm 1 (Encryption NBA 128 bit):
Input: plaintext 128 bit
Output: ciphertext 128 bit
Step 1: For i = 0-9
XL = XL XORP[i]
XR = CCSDPT(XL) XOR XR
Swap XL and XR
Step 2: Swap XL and XR //Undo the last swap//
Step 3: XR = XR XOR P[10]
Step 4: XL = XL XOR P[11]
Step 5: Recombine XL and XR

The CCS

Definition 1 (Cylinder coordinates): The cylindrical coordinates P(r, θ, z) of point P(r, θ, z) in xyz-space are shown geometrically in Fig. 3. The first two coordinates, r and θ, are the usual section polar coordinates except that r is regarded to be non-negative and r is restricted to the interval (0, 2π). The third coordinate is the third rectangular coordinate.

In rectangular coordinates, the coordinate surface $x = x_0$, $y = y_0$, $z = z_0$, consists of three mutually perpendicular sections. However, the coordinate surface assumes the following form (Salas *et al*., 2002; Hubbard and Hubbard, 2002): $r = r_0$, $\theta = \theta_0$, $z = z_0$.

Rectangular coordinates (x, y, z) can be obtained from cylindrical coordinates through Eq. 1:

$$x = r\cos\theta,\ y = r\sin\theta,\ z = z \tag{1}$$

Conversely, with the obvious exclusions, cylindrical coordinates (r, θ, z) can be obtained from rectangular coordinates (x, y, z) through Eq. 2:

$$r = \sqrt{x^2 + y^2}$$
$$\tan\theta = \frac{y}{x},\ z = z \tag{2}$$

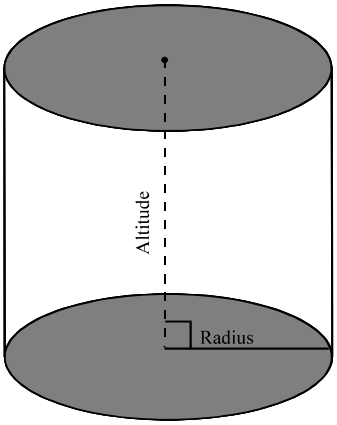


Fig. 4: Right cylindrical

From definition 1, researchers obtain the following properties:

- The cylindrical coordinates, defined in definition 1 are important when researchers have an axis symmetry line parallel to that of the axis
- The cylindrical coordinates map a point in space known by its attitude z, by the polar coordinates r, θ of its projects in the (x, y) section to a point in the (x, y, z) space

Definition 2 (The right cylinder): Let line:

$$x = x_0,\ y = y_0,\ z = z_0 \tag{3}$$

inspace $R^3 = R \times R \times R$. Researchers define the right cylinder as representing all points $P(x, y, z) \in R^3$ that satisfy $d(P, L) = r_0$.

Researchers designate r_0 as the radius of the cylinder and line L is the axis of cylinder. The right cylinder is a closed solid with two parallel (usually circular) bases connected by a curved surface as shown in Fig. 4 (Salas *et al*., 2002; Hubbard and Hubbard, 2002). The equation for the cylinder in the cylindrical coordinate is the axis of the right cylinder of $r = r_0$, $r^2\sin^2\theta + z^2 = r_0$ and $r^2\cos^2\theta + z^2 = r_0$ are the z-axis, y-axis and x-axis, respectively.

Definition 3 (Cylindrical section): Cylindrical sections are the intersections of cylinders with sections. For a right circular cylinder, four possibilities exist namely:

- Circle section
- Ellipse section
- Parabola section
- Hyperbola section

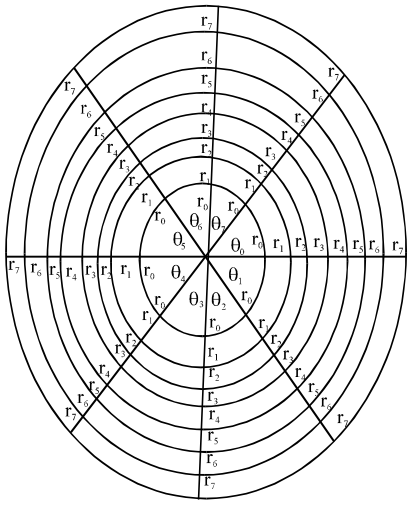


Fig. 5: Cross-section of the cylinder

In this study, researchers are interested in the circle section when itintersects the cylinder such that this sectionis parallel to the two bases. Thus, if the equation of the cylindrical coordinates is expressed as:

$$r = r_0,\ \alpha \le z \le b,\ 0 \le \theta < 2\pi \tag{4}$$

then, the equation of any section of the cylinder is:

$$z = z_0,\ 0 \le r \le r_0,\ 0 \le \theta \le 2\pi \tag{5}$$

This circular section contains numerous circles as shown in Fig. 5 and the equation of each section is:

$$r = r_s,\ 0 \le r_s \le r_0,\ 0 \le \theta \le 2\pi \tag{6}$$

The Mathematical Model: The right cylinder described in definition 2 is plotted in Fig. 6, into four parallel sections and converted into a matrix with three dimensions as follows:

$$A = [a_{ijk}]_{884}$$

where, a_{ijk} is related with point (r_i, θ_j, z_k) such that:

$$r_i \in \{0, 1, 2, ..., 8\}, i = 0, 1, 2, ...7$$

$$\theta_j \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{8}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\right\}, j = 0, 1, 2, ..., 7$$

$$z_k \in \{1, 2, ..., 4\}, k = 0, 1, 2, 3$$

The 3DS-box: The S-box hasan important role in the Block Cipher algorithm and is acritical step in any block cipher system. The S-boxes cause confusion in the enciphering process (El-Ramly *et al*., 2001) whereas permutation

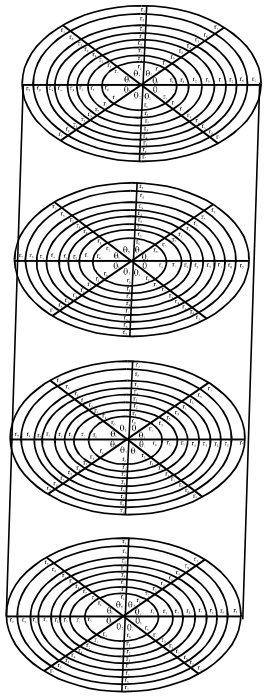


Fig. 6: Cylinder

causes diffusion of the security parameters (Naganathan *et al*., 2011). The 3D S-box in the CCSDPT is represented in 3D arrays of bytes A = $[a_{ijk}]_{8\times8\times4}$ constructed from the right cylinder as illustrated in the previous study. In study, the array in 3D is represented in the CCS where in each square (8×8 array) is shown as a set of 64 bytes that represents a section of the CCS for the right cylinder. Each row in the array represents one circle of the eight over lapping circle of the section in the right cylinder. Each individual byte in the section has three indices. The first index acts as a row number, the second index represents column number and the third index represents a section number. This setup allows any point in the right cylindrical coordinate to be referred to as a_{ijk}.

Representation of points for the right cylinder

Section 0 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$$r_0\ a_{000}\ a_{010}\ a_{020}\ a_{030}\ a_{040}\ a_{050}\ a_{060}\ a_{070}$$
$$r_1\ a_{100}\ a_{110}\ a_{120}\ a_{130}\ a_{140}\ a_{150}\ a_{160}\ a_{170}$$
$$r_2\ a_{200}\ a_{210}\ a_{220}\ a_{230}\ a_{240}\ a_{250}\ a_{260}\ a_{270}$$
$$r_3\ a_{300}\ a_{310}\ a_{320}\ a_{330}\ a_{340}\ a_{350}\ a_{360}\ a_{370}$$
$$r_4\ a_{400}\ a_{410}\ a_{420}\ a_{430}\ a_{440}\ a_{450}\ a_{460}\ a_{470}$$
$$r_5\ a_{500}\ a_{510}\ a_{520}\ a_{530}\ a_{540}\ a_{550}\ a_{560}\ a_{570}$$
$$r_6\ a_{600}\ a_{610}\ a_{620}\ a_{630}\ a_{640}\ a_{650}\ a_{660}\ a_{670}$$
$$r_7\ a_{700}\ a_{710}\ a_{720}\ a_{730}\ a_{740}\ a_{750}\ a_{760}\ a_{770}$$

Table 1: Comparison of the memory requirements of the S-boxes

Algorithms	Block size	Memory requirement (bytes)
BA	64	4096
Extension Blowfish	128	2,097,152
NBA 128 bit	128	256

Section 1 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$$r_0a_{001}a_{011}a_{021}a_{031}a_{041}a_{051}a_{061}a_{071}$$
$$r_1a_{101}\ a_{111}a_{121}a_{131}a_{141}a_{151}a_{161}a_{171}$$
$$r_2a_{201}a_{211}a_{221}a_{231}a_{241}a_{251}a_{261}a_{271}$$
$$r_3a_{301}a_{311}a_{321}a_{331}a_{341}a_{351}a_{361}a_{371}$$
$$r_4a_{401}a_{411}a_{421}a_{431}a_{441}a_{451}a_{461}a_{471}$$
$$r_5a_{501}a_{511}a_{521}a_{531}a_{541}a_{551}a_{561}a_{571}$$
$$r_6a_{601}a_{611}a_{621}a_{631}a_{641}a_{651}a_{661}a_{671}$$
$$r_7a_{701}a_{711}a_{721}a_{731}a_{741}a_{751}a_{761}a_{771}$$

Section 2 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$$r_0\ a_{002}\ a_{012}\ a_{022}\ a_{032}\ a_{042}\ a_{052}\ a_{062}\ a_{072}$$
$$r_1\ a_{102}\ a_{112}\ a_{122}\ a_{132}\ a_{142}\ a_{152}\ a_{162}\ a_{172}$$
$$r_2\ a_{202}\ a_{212}\ a_{222}\ a_{232}\ a_{242}\ a_{252}\ a_{262}\ a_{272}$$
$$r_3\ a_{302}\ a_{312}\ a_{322}\ a_{332}\ a_{342}\ a_{352}\ a_{362}\ a_{372}$$
$$r_4\ a_{402}\ a_{412}\ a_{422}\ a_{432}\ a_{442}\ a_{452}\ a_{462}\ a_{472}$$
$$r_5\ a_{502}\ a_{512}\ a_{522}\ a_{532}\ a_{542}\ a_{552}\ a_{562}\ a_{572}$$
$$r_6\ a_{602}\ a_{612}\ a_{622}\ a_{632}\ a_{642}\ a_{652}\ a_{662}\ a_{672}$$
$$r_7\ a_{702}\ a_{712}\ a_{722}\ a_{732}\ a_{742}\ a_{752}\ a_{762}\ a_{772}$$

Section 3 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$$r_0\ a_{003}\ a_{013}\ a_{023}\ a_{033}\ a_{043}\ a_{053}\ a_{063}\ a_{073}$$
$$r_1\ a_{103}\ a_{113}\ a_{123}\ a_{133}\ a_{143}\ a_{153}\ a_{163}\ a_{173}$$
$$r_2\ a_{203}\ a_{213}\ a_{223}\ a_{233}\ a_{243}\ a_{253}\ a_{263}\ a_{273}$$
$$r_3\ a_{303}\ a_{313}\ a_{323}\ a_{333}\ a_{343}\ a_{353}\ a_{363}\ a_{373}$$
$$r_4\ a_{403}\ a_{413}\ a_{423}\ a_{433}\ a_{443}\ a_{453}\ a_{463}a_{473}$$
$$r_5\ a_{503}\ a_{513}\ a_{523}\ a_{533}\ a_{543}\ a_{553}\ a_{563}\ a_{573}$$
$$r_6\ a_{603}\ a_{613}\ a_{623}\ a_{633}\ a_{643}\ a_{653}\ a_{663}\ a_{673}$$
$$r_7\ a_{703}\ a_{713}\ a_{723}\ a_{733}\ a_{743}\ a_{753}\ a_{763}\ a_{773}$$

The new F-function (CCSDPT): Researchers divided XL into four quarters (16 bit). In the BA, each quarter is used as an index for one of the S-boxes, there by requiring four S-boxes. By contrast, all quarters use the same 3D S-boxin the CCSDPT and byte permutation is applied to diffuse the elements after each quarter. Consequently, the number of S-boxes is reduced to one. Table 1 compares the memory requirements of the S-boxes of the BA, the extension Blow fish 128 bit and the NBA 128 bit.

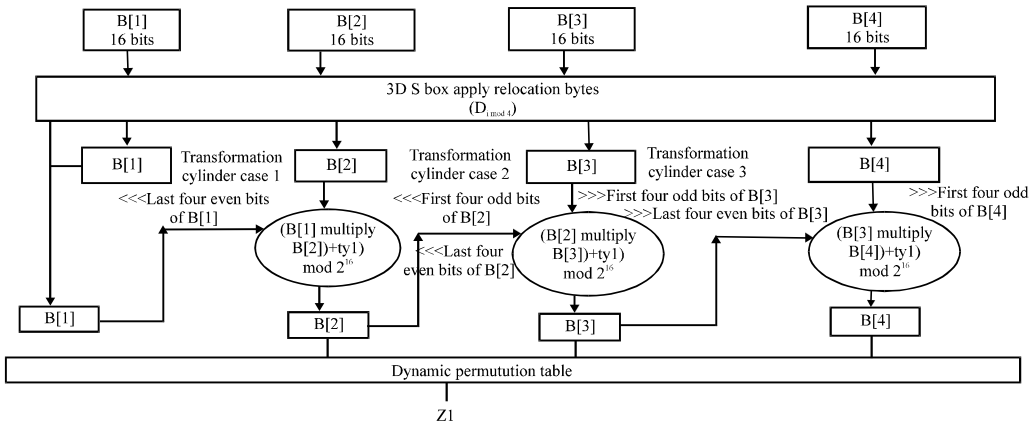


Fig.7: Diagram of CCSDPT

Each quarter (16 bit) in the CCSDPT is divided into two 8 bit parts. Each part (8 bit) is divided into three smaller parts represented as indexes to a_{ijk} where in the first three even bits of the byte are regarded as the index to the row number of a_{ijk} whereas the first three odd bits of the byte are regarded as the index to the column number of a_{ijk}. Hence, the last two bits of the byte are regarded as the index to the section number as described in Algorithm 2.

A complex function is required to eliminate the symmetries from the output of the 3D S-Box when different bytes of the input are equal. This complex function involves the use rotation, multiplication and addition with parts of the round subkey. Each quarter is multiplied by the previous quarter after rotating the former and added to part of the round sub key except for the first quarter. The result is then modulo 2^{16}. The part of the round key in each addition process is 16 bits. The multiplication of the second quarter is added to the first 16 bits from the round sub key (p[i]). The multiplication of the third quarter is added to the second 16 bits from p[i] and so on for the multiplication of the fourth quarter.

The number of rotations (right and left) is different for each output of the 3D S-box because this number depends on the four bits truncated from the output of the 3D S-box. Hence, if two or more bytes exist inthe 3D S-box have the same value in the 3D S-box then the results are not the same outputs. A diagram of the CCSDPT is shown in Fig. 7. Algorithm 2 of the new F-function (the CCSDPT) is as follows:

Algorithm 2 (The new F-function (the CCSDPT)):

Input: XL, subkeyround (p[i]), ith round
// XL is the left side of the plaintext 64 bit//
Output: z_1 //64-bis//
Step1: Divide XL into four 16 bit parts: B[1], B[2], B[3], B[4]
Step 2: Apply relocation bytes ($D_{i \bmod 4}$)
Step 3: Divide B[1] into two 8 bit parts: B1, B2
Set r = first three even bit of B1
Set c = first three odd bit of B1
Set p = two last bit of B1
Set r1 = first three even bit of B2
Set c1 = first three odd bit of B2
Set p1 = two last bit of B2
byte 1 = A [r, c, p]
byte 2 = A [r1, c1, p1]
B[1] = Combine byte 1 and byte 2
Step 4: Set j = 2
Step 5: While j≤4 do {the number of quarters}
Transformation of right cylinder bycase j-1
Divide B [j] into two 8 bit B1 and B2
r = first three even bits of B1
c = first three odd bits of B1
p = two last bits of B1
r1 = first three even bit of B2
c1 = first three odd bit of B2
p1 = two last bits of B2
byte 1 = A [r, c, p]
byte 2 = A [r1, c1,p1]
Combine byte 1 and byte 2 into byte 1_1
r = last four even bits of B[j-1]
c = first four odd bits of byte1_1
$Byte_{1\_2} = Byte_{1\_1} <<< r$
$Byte_{2\_2} = B[j-1] >>> c$
ty1 = subkey & 0×FFFF;
subkey = subkey>>16;
B[j] = (Byte 1_2×Byte 2 _2)+ty1) mod 2^{16}
j = j+1
End do
Step 6: Permute 8 bytes depending on the value of the permutation table

Relocation bytes: Relocation byte (D), the first procedure in byte permutation is based on secret keys applied in alternate rounds with the 3D S-box.

This procedure aims to vary the content of the 3D S-box variable for each round of the encryption process by using an approach explained in this section. Byte relocation involves four procedures, called D_0, D_1, D_2 and D_3.

D_0 affects one section of the CCS (the 3D S-box) for the right cylinder in the first round. D_1 performs the same function in the second round, D_2 in the third round and D_3 in the fourth round. These steps are looped repeatedly from D_0 ... D_3 until the 10 rounds are completed. D_0 ... D_3 always work on the 3D S-box generated from the key expansion part of the algorithm.

Researchers divide relocation bytes process into two steps. The first step involves choosing two of the four sections, depending on the secret random keys generated from the random function. This process is illustrated. The second step involves replacing the elements between the selected sections, depending on the other random secret keys generated from the same random function. Briefly, researchers generate five sets of random keys from the random function per round. Through this process, researchers benefit from the symmetry properties of the sections of the right cylinder. However, researchers divide eight circles per section into four quarters with each determined by a number in one of the other four sets that is, $s_n = \{k_{nm}, m = 0, ..., 3\}$ for $n = 1, ..., 4$. Researchers can then swap the elements in one quarter with the elements in another quarter.

Relocation keys: Relocation keys refer to the five sets of random numbers generated from the random function in interval [0, 3]. The seed of the random function is (XL XOR P[i])+sequence of block of plaintext). This process allows the random function to generate different random keys along the encryption process because its seed depends on the changing sequence of the plaintext block. $S_{n+1} = \{k_{nm} m = 0, 1, 2, 3\}$ for $n = 0, 1, 2, 3, 4$ which are generated randomly from the random function. Thus:

$$S_1 = \{k_{00}, k_{01}, k_{02}, k_{03}\}$$
$$S_2 = \{k_{10}, k_{11}, k_{12}, k_{13}\}$$
$$S_3 = \{k_{20}, k_{21}, k_{22}, k_{23}\}$$
$$S_4 = \{k_{30}, k_{31}, k_{32}, k_{33}\}$$
$$S_5 = \{k_{40}, k_{41}, k_{42}, k_{43}\}$$

The first set of the relocation keys is S_1 which represents the random four sections of the right cylinder. Researchers use the numbers in this set in the first step of relocation to select and pair up two sections. For example, if researchers have the numbers 1, 2, 0, 3 in the first set then researchers choose the second and third sections together and the first and fourth sections together.

The last four sets from the relocation keys specify the second step of the relocation byte that is, to swap the elements in the quarters among the selected sections. Table 2 illustrates examples of random numbers from the five sets.

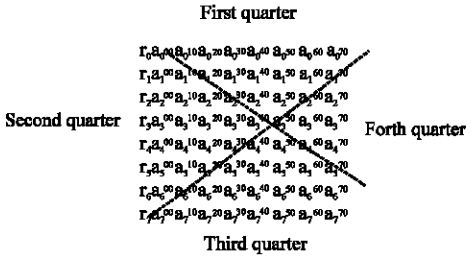
Table 2: Random numbers from the five sets (the relocation keys)

Set No.	Set	Representation
S_1	0213	Section number
S_2	1203	Quarter numbers of the first section
S_3	3102	Quarter numbers of the second section
S_4	1320	Quarter numbers of the third section
S_5	1032	Quarter numbers of the fourth section

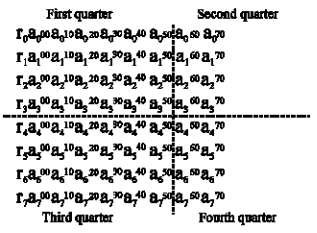
The numbers in the second set represent the quarter numbers in the first section, the numbers in the third set represents the quarter numbers in the second section and the numbers in the fifth set represent the quarter numbers in the fourth section. However, based on these sets and as shown in the preceding examples, researchers replace the elements in the second quarter of the first section with the elements in the second quarter of the third section, the elements in the third quarter of the first section with the elements in the fourth quarter of the third section, the elements in the first quarter of the first section with the elements in the third quarter of the third section and the elements in the fourth quarter of the first section with the elements in the first quarter of the third section. The same procedure is performed with the second and the fourth sections, depending on their sets. The division into quarters is different for each relocation byte (D_0, D_1, D_2 and D_3). D_0, D_1, D_2 and D_3 work on the sections of the 3D S-box generated from the key expansion part of the algorithm.

The division of the sections with each relocation byte on the first section ($section_0$) is illustrated in the study.

Quarters in the first section with D_0, D_1, D_2, and D_3
Section 0 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):



Section 0 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):



Section 0 (8×8) bytes $\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$:

$r_0 a_{000}\ a_{010}\ a_{020}\ a_{030}$	$a_{040}\ a_{050}$	$a_{060}\ a_{070}$
$r_1 a_{100}\ a_{110}\ a_{120}\ a_{130}$	$a_{140}\ a_{150}$	$a_{160}\ a_{170}$
$r_2 a_{200}\ a_{210}\ a_{220}\ a_{230}$	$a_{240}\ a_{250}$	$a_{260}\ a_{270}$
$r_3 a_{300}\ a_{310}\ a_{320}\ a_{330}$	$a_{340}\ a_{350}$	$a_{360}\ a_{370}$
$r_4 a_{400}\ a_{410}\ a_{420}\ a_{430}$	$a_{440}\ a_{450}$	$a_{460}\ a_{470}$
$r_5 a_{500}\ a_{510}\ a_{520}\ a_{530}$	$a_{540}\ a_{550}$	$a_{560}\ a_{570}$
$r_6 a_{600}\ a_{610}\ a_{620}\ a_{630}$	$a_{640}\ a_{650}$	$a_{660}\ a_{670}$
$r_7 a_{700}\ a_{710}\ a_{720}\ a_{730}$	$a_{740}\ a_{750}$	$a_{760}\ a_{770}$
First quarter	Second quarter	Third quarter	Fourth quarter

Section 0 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$r_0 a_{000} a_{010} a_{020} a_{030} a_{040} a_{050} a_{060} a_{070}$
$r_1 a_{100} a_{110} a_{120} a_{130} a_{140} a_{150} a_{160} a_{170}$ First quarter
---
$r_2 a_{200} a_{210} a_{220} a_{230} a_{240} a_{250} a_{260} a_{270}$
$r_3 a_{300} a_{310} a_{320} a_{330} a_{340} a_{350} a_{360} a_{370}$ Second quarter
---
$r_4 a_{400} a_{410} a_{420} a_{430} a_{440} a_{450} a_{460} a_{470}$
$r_5 a_{500} a_{510} a_{520} a_{530} a_{540} a_{550} a_{560} a_{570}$ Third quarter
---
$r_6 a_{600} a_{610} a_{620} a_{630} a_{640} a_{650} a_{660} a_{670}$
$r_7 a_{700} a_{710} a_{720} a_{730} a_{740} a_{750} a_{760} a_{770}$ Fourth quarter

Algorithm 3 of the relocation bytes is as follows:

Algorithm 3 relocation bytes:

Input: $section_0$, $section_1$, $section_2$, $section_3$ of the 3D S-box in the key expansion part of the algorithm, five sets of random relocation keys from the random function, ith round

Output: $section_0$, $section_1$, $section_2$, $section_3$ after applying there location bytes

Step 1: Divide each section into four quarters based on $D_{(imod 4)}$

Step 2: Select a pair of sections depending on the generated random keys of the first set

Step 3: Swap the elements of the quarters among selected sections in step 2, depending on the other four sets

Note: The D_0 swaps between the elements in the main diagonals with the elements in the second diagonals of the selected sections are as follows:

Swap (L_0, L_1),
Swap (L_2, L_3),
Swap (S_0, S_2),
Swap (S_1,S_3).

{L_0, L_1, L_2, and L_3 are the main diagonals and S_0, S_1, S_2 and S_3 are the second diagonals in the selected sections}

Transformations in the right cylinder: The second procedure of a permutation byte is called the transformation of the right cylinder coordinate which involves rotation and translation, depending on the following transformations:

$$T(r_i, \theta_j, z_k) = \left((r_i \pm r) \bmod 8, (\theta_j \pm \theta) \bmod 2\pi, (z_k \pm z) \bmod 4\right)$$

Such that:

$$\theta = \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{8}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\right\}$$

$r = \{0, 1, 2, ..., 7\}$

$z = \{0,1,2,3\}$

$i, j = \{0, 1, ..., 7\}$

$k = \{0, 1, 2, 3\}$

where, θ, z are random secret keys. After taking every two bytes (16 bit) from the sections of the right cylinder (the 3D S-box), the right cylinder becomes a transformed module implementation. The implementation consists of three cases. Thus, after the first two bytes are obtained from the 3D S-box, the first case is implemented after the second two bytes are obtained, the second caseis implemented and after the third two bytes are obtained, the third case is implemented. Thus, researchers apply the second case on the resulting 3D S-box from the first case, the third case on the resulting 3D S-box from the second case and the first case on the resulting 3D S-box from the relocation byte.

Researchers use these transformations to diffuse the elements of the 3D S-box after the 2 bytes are produced. Researchers select the first three cases from the eight cases that are used in the transformation. Researchers can choose any of the other cases:

- Case $\theta \neq 0$
- $Case_1$ $r \neq 0$, $z \neq 0$
- $Case_2$ $r \neq 0$, $z = 0$
- $Case_3$ $r = 0$, $z \neq 0$
- $Case_4$ $r = 0$, $z = 0$
- Case $\theta = 0$
- $Case_5$ $r = 0$, $z = 0$
- $Case_6$ $r = 0$, $z \neq 0$
- $Case_7$ $r \neq 0$, $z = 0$
- $Case_8$ $r \neq 0$, $z \neq 0$

For example: a cycleis obtained and rotated by $\theta = \pi/4$ as shown in Fig. 8. This rotation $(\theta_j \pm \theta) \bmod 2\pi$ on the first section is applied only when θ is $\pi/4$. The rotation of the first section by $\theta = \pi/4$ is shown in study.

Rotation of the first section by $\theta = \pi/4$
Section 0 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$r_0 a_{070} a_{000} a_{010} a_{020} a_{030} a_{040} a_{050} a_{060}$
$r_1 a_{170} a_{100} a_{110} a_{120} a_{130} a_{040} a_{150} a_{160}$
$r_2 a_{270} a_{200} a_{210} a_{220} a_{230} a_{240} a_{250} a_{260}$
$r_3 a_{370} a_{300} a_{310} a_{320} a_{330} a_{340} a_{350} a_{360}$
$r_4 a_{470} a_{400} a_{410} a_{420} a_{430} a_{440} a_{450} a_{460}$
$r_5 a_{570} a_{500} a_{510} a_{520} a_{530} a_{540} a_{550} a_{560}$
$r_6 a_{670} a_{600} a_{610} a_{620} a_{630} a_{640} a_{650} a_{660}$
$r_7 a_{770} a_{700} a_{710} a_{720} a_{730} a_{740} a_{750} a_{760}$

The translation $[(r_i \pm r) \bmod 8]$of the first section, where r = 2.

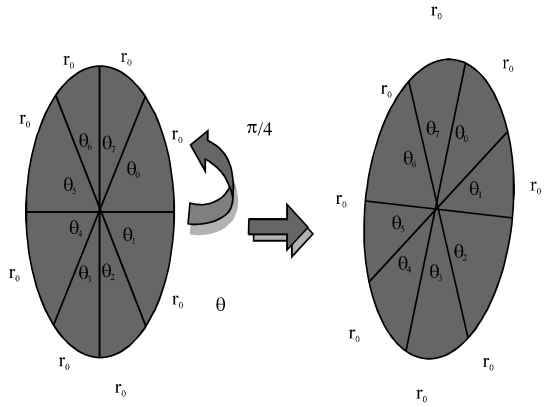


Fig. 8: Rotating the cycle by = π/4

Translation of the first section by r = 2
Section 0 (8×8) bytes ($\theta_0\theta_1\theta_2\theta_3\theta_4\theta_5\theta_6\theta_7$):

$$r_0 a_{600} a_{610} a_{620} a_{630} a_{640} a_{650} a_{660} a_{670}$$
$$r_1 a_{700} a_{710} a_{720} a_{730} a_{740} a_{750} a_{760} a_{770}$$
$$r_2 a_{000} a_{010} a_{020} a_{030} a_{040} a_{050} a_{060} a_{070}$$
$$r_3 a_{100} a_{110} a_{120} a_{130} a_{140} a_{150} a_{160} a_{170}$$
$$r_4 a_{200} a_{210} a_{220} a_{230} a_{240} a_{250} a_{260} a_{270}$$
$$r_5 a_{300} a_{310} a_{320} a_{330} a_{340} a_{350} a_{360} a_{370}$$
$$r_6 a_{400} a_{410} a_{420} a_{430} a_{440} a_{450} a_{460} a_{470}$$
$$r_7 a_{500} a_{510} a_{520} a_{530} a_{540} a_{550} a_{560} a_{570}$$

While translating $[(z_k \pm z) \bmod 4]$ on the cylinder, where z = 2, the first section becomes the third section, the second section becomes the fourth section, the third section becomes the first section and the fourth section becomes the second section.

The DPT: Researchers use the DPT in this step to diffuse the outputs of the 3D S-box (8 bytes). The permutation table is changeable in each round ofthe encryption process. This tendency limits the opportunity for differential and linear cryptanalyses. Fixed permutation tables, similar to those in the DES P-boxes, are designed to diffuse the output bit alteration from the S-boxes thus providing an opportunity for differential and linear crypt analyses.

Researchers explain the steps for the DPT based on the secret keys in Algorithm 4. Researchers design this table based on the two different secret keys: the first key is 6 even bits from the subkey round (p[i]) and the second key is 6 bits from the outputs of the 3D S-Box (8 bytes). The initial values entered into the permutation table range from 0-63. Then, researchers compute the index (n) of the permutation table which depends on these secret keys by using Eq. 7:

$$n = (n + permutation\_table[k]) + ty) \bmod 64 \quad (7)$$

Where:
n = The first secret key
ty = The second secret key

Then, the value of index (n) is swapped with a value in the permutation table. This process is repeated 64 times. In each time, researchers compute a new value for the second secret key by using another 6 bits from the 8 bytes. This cycle is repeated through the 8 bytes (tx) and a new index (n) is computed by using Eq. 7. The value of the index (n) and index (k) in the premutation table are then swapped. The outputs of the 3D S-box (8 bytes) are permuted based on the values in the premutation table. Algorithm 4 of the dynamic permutation in the NBA 128-bit is as follows:

Algorithm 4: DPT

```
Input: 8 bytes, subkey round(p[i])
Output: 8 bytes after diffusion
Step 1: Set n = first 6 even bits of the round subkey
Step 2: Set tx = 8 bytes
Step 3: Initial permutation table with values of
        0,...,63
  for k= 0-63
permutation_table [k] = k
Step 4:For k = 0-63
          begin
ty = tx&0×3F //first six bit from tx//
n = (n+permutation_table[k])+ty)mod 64
Swap (permutation_table[n], permutation_table[k])
tx = tx>>6
ty = ty<<(64-6)
tx = tx|ty
  //take the second 6 bits from tx and repeat the cycle through the tx until the
end of loop//
      End
   Step 5: Permutation of 8 bytes based on the values in the
           permutation table
```

Data decryption: The procedure of the Decryption algorithm of a block cipher should be identical to that of the Encryption algorithm but inthe reverse order. The Encryption algorithm procedure is designed such that the Decryption algorithm procedure becomes identical to that of the encryption algorithm in the same order except for the application of the subkeys which is performed in reverse order.

Algorithm 5 (Decryption NBA 128 bit):

```
Input: Ciphertext 128 bit
   Output: Plaintext 128 bit
Step 1:XL = XL XOR P[11]
Step 2:XR = XR XOR P[10]
Step 3:For i = 9 to 0
          XR = CCSDPT (XL) XOR XR
           XL =  XL XOR P[i]
       SWAP XL and XR
Step 4: SWAP XL and XR
// to undo the last swap //
Step 5: Recombine XL and XR to obtain the plaintext
```

SECURITY ANALYSIS

Security is the most important factor in evaluating Cryptographic algorithms. Security includes features such as the randomness of the algorithm output, the avalanche effect, the correlation coefficient, the resistance of the algorithm to the crypt analysis and the relative security compared with other candidates (Ariffin, 2012). In this study, security analysis includes analysis the randomness of the RAF and the results are compared with those of the BA.

Randomness analysis of the 128 bit NBA: The statistical tests aim to measure the randomness quality of a generator and to detect its weaknesses. The NIST test Suite (Bassham *et al*., 2010) is a set of statistical tests that are commonly used to determine whether the binary sequence possesses specific characteristics that a truly random sequence will likely exhibit. This suite consists of 15 tests developed to analyze the randomness of arbitrarily long binary sequences produced by either a hardware or software-based cryptographic random number generator. Table 3 shows the list of the 188 statistical tests conducted in the experiments.

Testing data: In this study, researchers provide five types of testing data used with the NIST test on the BA and the NBA 128 bit. A number of these testing data such as Cipher Block Chaining (CBC) mode and random plaintext/random 128 bit keys are used to select the final candidates for the Advanced Encryption Standard (AES) block cipher by Soto and Bassham (2000) and Soto (1999) Therefore, researchers use the same data and sample size in this study. Three other types of testing data (image, text and video) used to evaluate the BA by Chandrasekaran *et al*. (2011), Elminaam *et al*. (2010), Mousa (2005), Meyers and Desoky (2008), Suri and Deora (2010) and Mandal (2012) are also used in this study.

Table 3: The 188 statistics tests conductedinthe experiments

Statistical test	No. of p-values	Test ID
Frequency	1	1
Block frequency	1	2
Cumulative sum	2	3-4
Runs	1	5
Longest run	1	6
Rank	1	7
FFT	1	8
Non-overlapping template	148	9-156
Overlapping template	1	157
Universal	1	158
Approximate entropy	1	159
Random excursions	8	160-167
Random excursion variant	18	168-185
Serial	2	186-187
Linear complexity	1	188

CBC mode: Given a random 128 bit key, a 128 bit Initialization Vector (IV) of all zeroes and a 128 bit Plain Text block (PT) of all zeroes, a binary sequence of 1,048,576 bits can then be constructed by using the cipher text computed in the CBC mode that is a binary sequence that consists of 8192 concatenated 128 bit cipher text blocks. The first Cipher Text block (CT1) is defined by CT1 = Ek(IV⊕PT). Subsequent cipher text blocks are defined by CTi+1 = Ek(CTi⊕PT) for $1 \leq i \leq 8192$. In total, 300 binary sequences are constructed each with a different random 128 bit key (Soto and Bassham, 2000; Soto, 1999). These data are used withthe NBA 128 bit. The same type of data is reused with the BA but the number of blocks is 16384 instead of 8192 and the length of the block is 64 bit.

The random plaintext/random 128 bit keys: The random plaintext/random 128 bit keys are based on the data generated using the Blum-Blum-Shub (BBS) pseudorandom bit generator. The BBS is selected in this experiment because it seems to be a cryptographically secure pseudo-random bit generator (Menezes *et al*., 1997) and it is of the same data type used totest the AES final candidates (Soto and Bassham, 2000; Soto, 1999). A total of 128 sequences are constructed to examine the randomness of the ciphertext (based on the random plaintext and the random 128 bit keys). Each sequence results from the concatenation of 8128 cipher text blocks of 128 bitlength (1,040,384 bits) using 8128 random plaintext blocks of 128 bit length and a random 128 bit key. These data are used in theNBA 128 bit. The same type of data is reused in the BA but the number of blocks is 16256 instead of 8128 and the length of the block is 64 bit. The BBS is then implemented by using Java language (NetBeans IDE 7.2).

Image files: The data set has 128 sequences of image files in different formats. Each sequence is a result of the concatenation of 12290 (1,573,120 bits) ciphertext blocks of 128 bitlength using 12290 plaintext blocks of 128 bitlength and a random 256 bit key. These data are used in the NBA 128 bit. The same type of data is reused in the BA but the number of blocks is 24580 instead of 12290 and the length of the block is 64 bit.

Text files: The data set has 128 sequences of text files. Each sequence results from the concatenation of 8128 (1,040,384 bits) ciphertext block of 128 bitlength using 8128 plaintext blocks of 128 bitlength and a random 256 bit key. These data are used in the NBA 128 bit. The same type of the data is reused in the BA but the number of blocks is 16256 instead of 8128 and the length of the block is 64 bit.

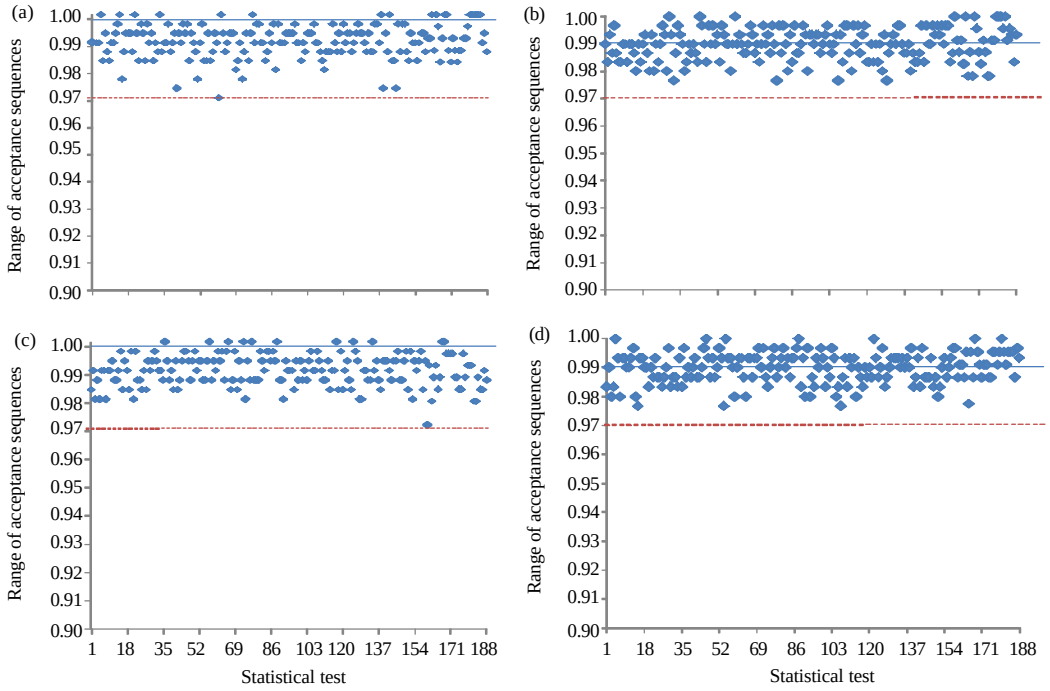


Fig. 9: Results of cipher block chaining mode; a) Blowfish 64 round 2 (Cipher block chaining mode); b) Blowfish 64 round 4 (Cipher block chaining mode); c) NBA 128 round 2 (Cipher block chaining mode) and d) NBA 128 Round 4 (Cipher block chaining mode)

Video files: The data set has 128 sequences of video files. Each sequence results from the concatenation of 8128 (1,040,384 bits) cipher text block of 128 bitlength using 8128 plaintext blocks of 128 bitlength and a random 256 bit key. These data are used in the NBA 128 bit. The same type of data is reused in the BA but the number of blocks is 16256 instead of 8128 and the length of the block is 64 bit.

Empirical results and analysis: The randomness testing is conducted in full and partial rounds of the BA, the extended 128 bit BA and the NBA 128 bit. Full Round Testing (FRT) and Partial Round Testing (PRT) are explained in the study.

FRT: In FRT, all five data types are generated using a full round of the BA which means that the derived data must complete 16 rounds of the BA and 10 rounds of the NBA 128 bit.

PRT: Soto and Bassham (2000) tested two fish rounds in pairs. Twofish is a Feistel network. Each round leaves some of the data bit unchanged and thus, Twofish will not appear as random under the test conditions after one round. However, all data bits are affected after two rounds. Twofish rounds are evaluated in pairs, namely, the even-numbered rounds from 2-14. Therefore, during the PRT of the BA, all five data types are generated by using the partial round of the BA in pairs from 2-14 for the BA where as rounds 2-8 are used for the NBA 128 bit.

The test on the CBCmode for the two pairs of rounds of the BA and the NBA 128 bit is illustrated in Fig. 9. In Fig. 9, the dashed line depicts the smallest proportion that satisfies the 0.01 acceptance criterion whereas the solid line depicts the expected proportion. The results clearly show that the outputs from both algorithms are random by the end of the second round (the first round pair) because majority of the 188 statistical tests fall above 97%. Subsequent rounds produce similar statistics.

The test on random plaintext/random 128 bit for the two pairs of rounds of the BA and the NBA 128 bitis illustrated in Fig. 10. The results clearly show that the outputs from both algorithms are random by the end of the second round (the first round pair) because majority of the 188 statistical tests fall above 96%. Subsequent rounds produce similar statistics.

The test on image files for the two pairs of rounds of the BA and the NBA 128 bit is illustrated in Fig. 11. The results clearly show that the output from the BA remains non-random by the end of the second round (the first round pair) because majority of the 188 statistical tests fall below 96%. During the fourth round (the second round

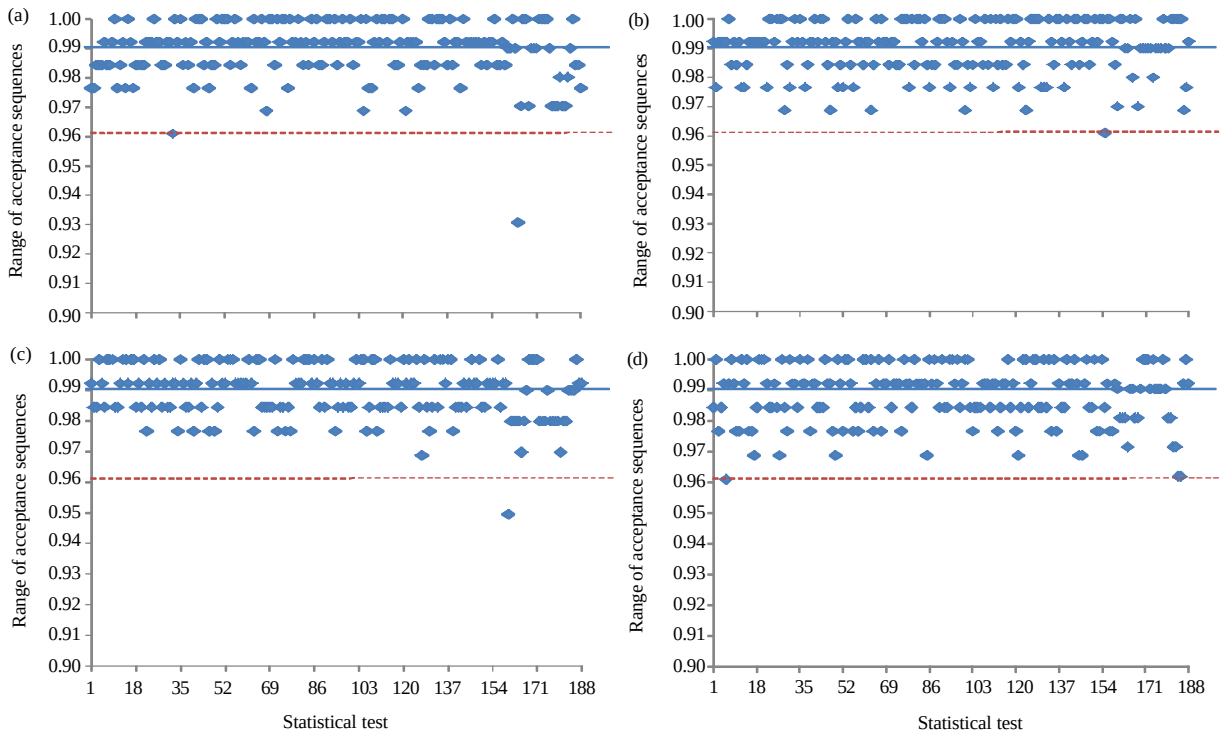


Fig. 10: Results of the random plaintext/random 128 bit keys; a) Blowfish 64 round 2 (bbs); b) Blowfish 64 round 4 (bbs); c) NBA 128 round 2 (bbs) and d) NBA 128 round 4 (bbs)

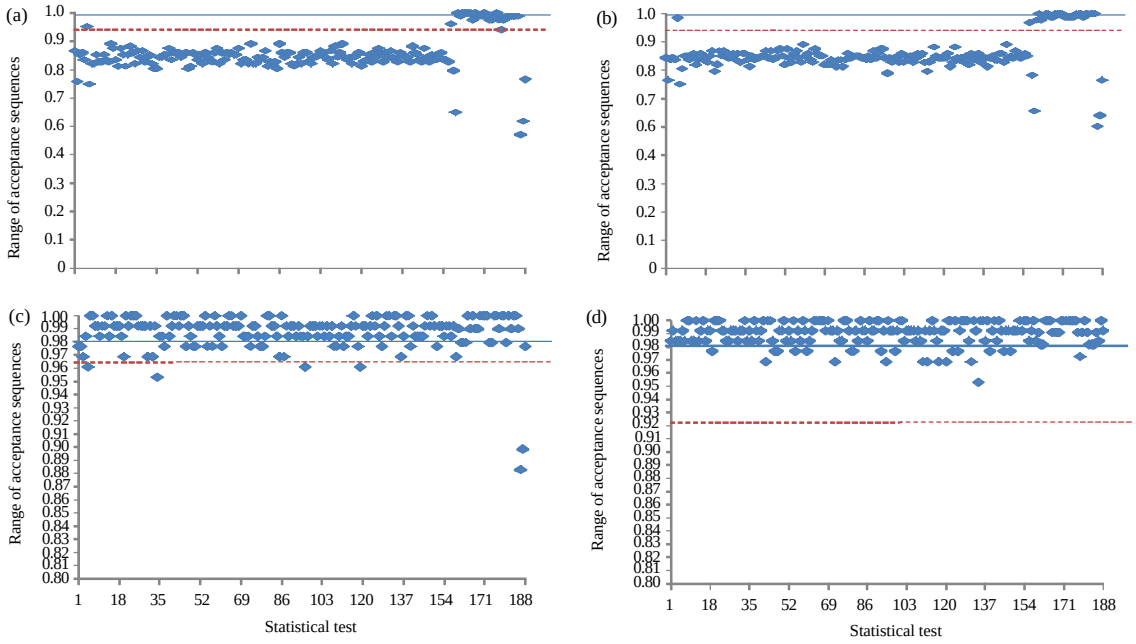


Fig. 11: Results of image files; a) Blowfish 64 round 2 (image); b) Blowfish 64 round 4 (image); c) NBA 128 round 2 (image) and d) NBA 128 round 4 (bbs)

pair) the output from the BAremains non-random because majority of the 188 statistical tests fall below 96%. Subsequent rounds produce similar statistics. Meanwhile, the output from the NBA 128 bit remains non-random by

the end of the second round (the first round pair) because majority of the 188 statistical tests fall below 96%. However, the output from the NBA 128 bit becomes random by the end of the fourth round (the second round pair) because majority of the 188 statistical tests fall above 96%. Subsequent rounds produce similar statistics. The test on text files for the two pairs of rounds of the BA and the NBA 128 bitis illustrated in Fig. 12. The results clearly show that the output from the BA remains non-random by the end of the second round (the first round pair) because

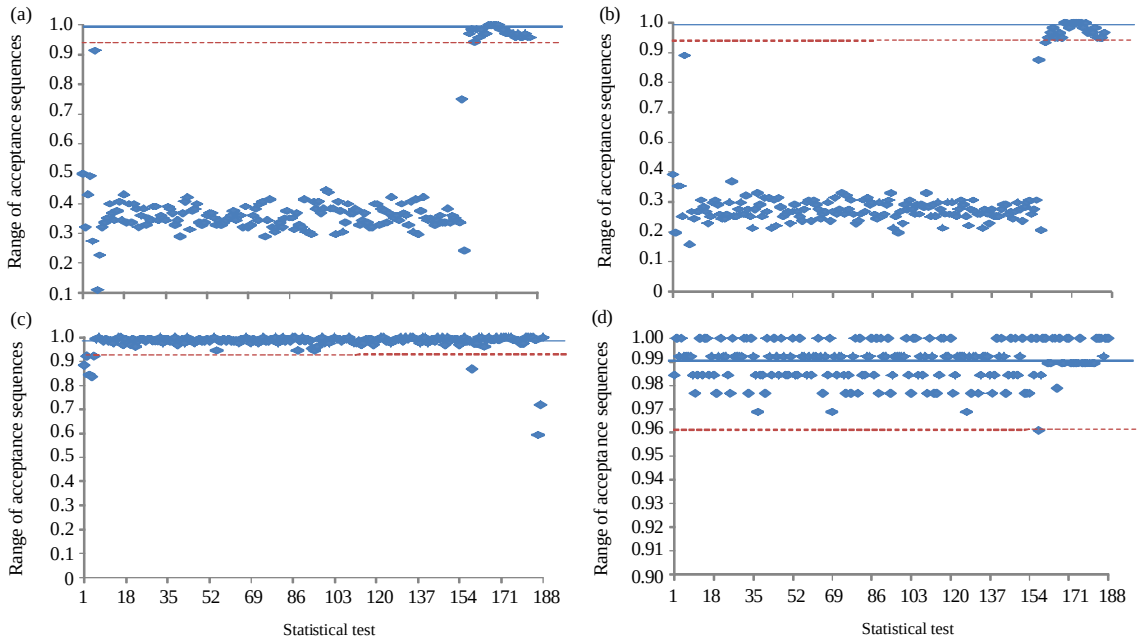


Fig. 12: Results of text files; a) Blowfish 64 round 2 (text); b) Blowfish 64 round 4 text); c) NBA 128 round 2 (text) and d) NBA 128 round 4 (text)

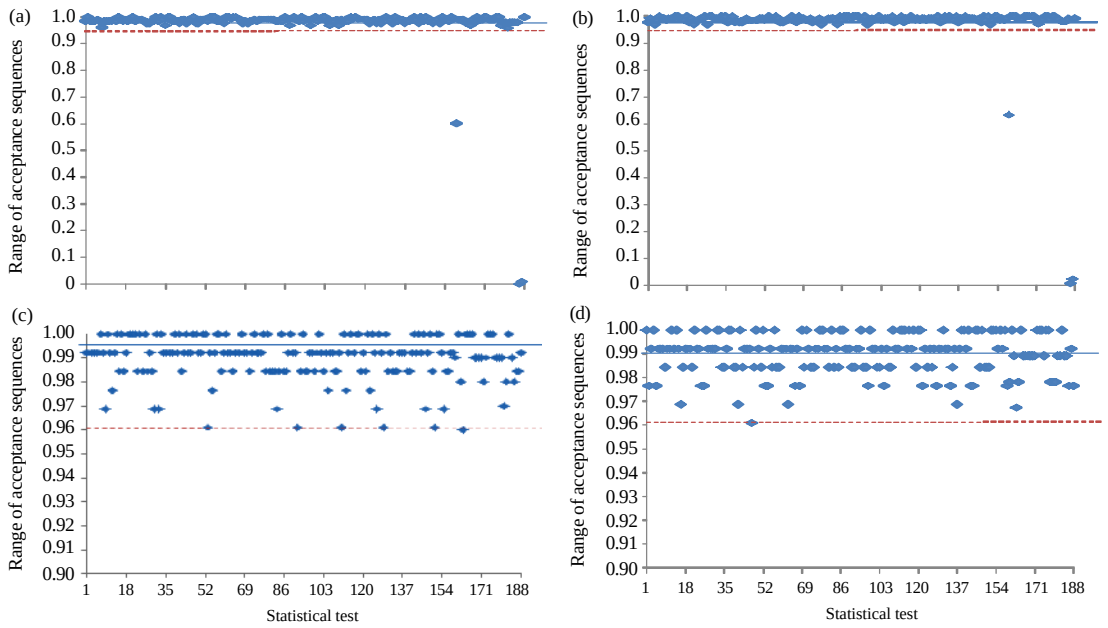


Fig. 13: Results of video files; a) Blowfish 64 round 2 (video); b) Blowfish 64 round 4 (video); c) NBA 128 round 2 (video) and d) NBA 128 round 4 (video)

majority of the 188 statistical tests fall below 96%. During the fourth round (the second round pair)), the output from the Ba remains non-random because majority of the 188 statistical tests fall below 96%. Subsequent rounds produce similar statistics. Meanwhile, the output from the NBA 128 bit remains nonrandom by the end of the second round (the first round pair) because majority of the 188 statistical tests fall below 96%. The output from the NBA 128 bit becomes random by the end of the fourth round (the second round pair) because majority of the 188 statistical tests fall above 96%. Subsequent rounds produce similar statistics.

The test on video files for the two pairs of rounds of the BA and the NBA 128 bitis illustrated in Fig. 13. The results clearly show that the output from the BA is random at the end of the second round (the first round pair) because majority of the 188 statistical tests fall above 96% except approximate entropy (159) and serial (186, 187) statistical test which fall below 96%. Subsequent rounds produce similar statistics. The output fromthe NBA 128 bit is also random by the end of the second round (the first round pair) because majority of the 188 statistical tests fall above 96%. Subsequent rounds produce similar statistics.

CONCLUSION

The proposed algorithm has been implemented and tested during the design process to obtain the experimental results. Several conclusions are drawn from this research and the most significant ones are discussed as follows. The function of the CCSDPT which is based on a CCS with aDPT as well as on multiple secret keys provide high resistance against differential and linear attacks. Hence, this technique is better than the BA because the 3D S-box and the permutation box in the NBA 128 bit are not only fixed entities but can also frustrate any attempt from attackers including the construction of iterative linear and differential relations. Based on the results of the NIST tests, the BA is unsuitable for image and text files with large strings of identical bytes. Meanwhile, the NBA 128 bit is designed to encrypt all file formats without restrictions on the contents of the files even if they have a large string of identical bytes because this approach is based on the sequence of the plaintext as the secret key.

The CCSDPT function is based on the DPT and the dynamic 3D S-box based on multiple secret keys that are changeable during the encryption process. These secret keys are generated from the random function with numerous variables, one of which is the sequence of the plaintext. These features do not only strengthen the security of the generated S-box but also makes the box compatible with any type of data even long string identical data.

RECOMMENDATIONS

Following the present study, future research can be conducted on the following topics:

- Analyzing the performance of the NBA 128 bit based on following factors: a valanche affect and correlation coefficient
- Analyzing 3D S-box based on criteria S-box
- Analyzing crypt analysis of the attacks on NBA 128 bit
- Recommending the development of a new procedure to generate 3D S-box and p-array in the NBA 128 bit for a new CCSDPT function

REFERENCES

Al-Neaimi, A.M.A. and R.F. Hassan, 2011. New approach for modified blowfish algorithm using 4-states keys. Int. J. Comput. Sci. Network Secur., 11: 21-26.

Alabaichi, A., R. Mahmod and F. Ahmad, 2013a. Randomness analysis of 128 bits blowfish block cipher on ECB mode. Int. J. Comput. Sci. Inform. Secur., 11: 8-21.

Alabaichi, A.M., R. Mahmod, M.S. Mechee and F. Ahmad, 2013b. Randomness analysis on blowfish block cipher using ECB and CBC Modes. Applied Sci. J., 13: 768-789.

Ariffin, S., 2012. A human immune system inspired byte permutation of block cipher. Ph.D. Thesis, Universiti Putara Malaysia, Malaysia.

Bassham, L.E., A.L. Rukhin, J. Soto, J. R. Nechvatal and M.E. Smid *et al* ., 2010. A statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology, Technical Report No. SP 800-22, 2010.

Chandrasekaran, J., B. Subramanyan and G.S. Raman, 2011. Ensemble of blowfish with chaos based s box design for text and image encryption. Int. J. Network Secu. Appl., 3: 165-173.

Cornwell, J.W., 2012. Blowfish survey. https://cs.columbusstate.edu/cae-ia/StudentPapers/cornwell.jason.pdf.

El-Ramly, S. H., T. El-Garf and A.H. Soliman, 2001. Dynamic generation of S-boxes in block cipher systems. Proceedings of the IEEE Eighteenth National Radio Science Conference, March 27-29, 2001, Mansoura, pp: 389-397.

Elminaam, D.S.A., H.M. Abdual-Kader and M.M. Hadhoud, 2010. Evaluating the performance of symmetric encryption algorithms. Int. J. Network Sec., 10: 216-222.

Hashim, A.T., S.M. Al-Qarrawy and J.A. Mahdi, 2009. Design and Implementation of an Improvement of Blowfish Encryption Algorithm. IJCCCE, 9: 1-15.

Hubbard, J.H. and B.B. Hubbard, 2002. Vector Calculus, Linear Algebra, and Differential Forms: A Unified Approach. 2nd Edn., Prentice Hall, USA., ISBN: 9780130414083, Pages: 800.

Mahdi, J.A., 2009. Design and implementation of proposed B-R encryption algorithm. IJCCCSE, 9: 1-17.

Mandal, P.C., 2012. Evaluation of performance of the symmetric key algorithms: DES, 3DES, AES and blowfish. J. Global Res. Comput. Sci., 3: 67-70.

Manikandan, G., P. Rajendiran, K. Chakarapani, G. Krishnan and G. Sundarganesh, 2012a. A modified crypto scheme for enhancing data security. J. Theor. Applied Inform. Technol., 35: 149-154.

Manikandan, G., N. Sairam and M. Kamarasan, 2012b. A new approach for improving data security using iterative blowfish algorithm. Res. J. Applied Sci., 4: 603-607.

Menezes, A.J., P.C. van Oorschot and S.A. Vanstone, 1997. Handbook of Applied Cryptography. CRC Press, New York, USA.

Meyers, R.K. and A.H. Desoky, 2008. An implementation of the blowfish cryptosystem. Proceedings of the IEEE International Symposium, Signal Processing and Information Technology, December 16-19, 2008, Sarajevo, Bosnia and Herzegovina, pp: 346-351.

Mousa, A., 2005. Data encryption performance based on Blowfish. Proceedings of the 47th International Symposium ELMAR, June 8-10, 2005, Zadar, Croatia, pp: 131-134.

Naganathan, E. R., V. Nandakumar and S.S. Dhenakaran, 2011. Identity based encryption using feistel cipher. Eur. J. Sci. Res., 54: 532-537.

Nechvatal, J., E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti and E. Roback, 2000. Report on the Development of the Advanced Encryption Standard (AES). National Institute of Standard and Technology, USA.

Nie, T. and T. Zhang, 2009. A study of DES and Blowfish encryption algorithm. Proceedings of the IEEE Region 10 Conference TENCON, January 23-26, 2009, Singapore, pp: 1 4.

Salas, S.L., J. Garret and E.E. Hille, 2002. Calculus: One and Several Variables. 9th Edn., John Wiley and Sons, New York.

Schneier, B., 1994. Description of a new variable-length key, 64-bit block cipher (Blowfish). Proceedings of the Cambridge Security Workshop, December 9-11, 1994, Cambridge, UK., pp: 191-204.

Shannon, C.E., 1949. Communication theory of secrecy systems. Bell Syst. Tech. J., 28: 656-715.

Soto, J. and L. Bassham, 2000. Randomness testing of the advanced encryption standard finalist candidates. National Institute of Standards and Technology, NIST IR 6483, pp: 1-14. http://csrc.nist. gov/ publications/nistir/ir6483.pdf.

Soto, J., 1999. Randomness testing of the advanced encryption standard candidate algorithms. U.S. Department Of Commerce, NISTIR 6390, September 1999. http://csrc.nist.gov/publications/nistir/ir6390. pdf.

Suri, P.R. and S.S. Deora, 2010. A cipher based on 3D array block rotation. Int. J. Comput. Sci. Network Secur., 10: 186-191.

Wang, Z., J. Graham, N. Ajam and H. Jiang, 2011. Design and optimization of hybrid MD5-blowfish encryption on GPUs. Proceedings of the International Conference on Parallel and Distribute Processing Techniques and Applications, July 18-21, 2011, Las Vegas, Nevada, USA.

Zhang, R. and L. Chen, 2008. A block cipher using key-dependent S-box and P-boxes. Proceedings of the IEEE International Symposium on Industrial Electronics, June 30-July-2, 2008, Cambridge, pp: 1463-1468.